UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/772,992	02/05/2004	James S. Miller	13768.493	5389

47973 7590 07/10/2008

WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

NOTIFICATION DATE	DELIVERY MODE
07/10/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

Office Action Summary	Application No.	Applicant(s)	
	10/772,992	MILLER ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *March 20, 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-23,26 and 27* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,4-23,26 and 27* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☒ Interview Summary (PTO-413) Paper No(s)/Mail Date. *4/3/2008* .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated March 20, 2008, responding to the Office action mailed November 23, 2007 provided in the rejection of claims 1-27, wherein claims 1, 4, 6-7, 10, 16, 20-23, and 26-27 were amended, claims 2-3 and 24-25 were canceled.

Claims 1, 4-23, and 26-27remain pending in the application and which have been fully considered by the examiner.

The status of claims 22 and 27 were inadvertently stated as "Previously Presented".

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Cook* - art made of record, as applied hereto.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Objections

2. Claim 4 is objected to because the following informalities:

- "The method as recited in claim 2 …", claim 2, line 1, should be corrected "The method as recited in claim 1 …"

Claim Rejections – 35 USC § 102(b)

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(b) that form the basis for the rejections under this section made in this office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

3. Claims 1, 4-17, 20-22, and 26-27 are rejected under 35 U.S.C. 102(b) as being anticipated by Jonathan E. Cook (*Supporting Rapid Prototyping through Frequent and Reliable Deployment of Evolving Components, 2001 IEEE, pp. 194-199*) (hereinafter 'Cook' - art made of record)

4. **As to claim 1** (Currently Amended), Cook discloses in a computerized system that includes one or more program computer-executable program components including one or more computer-executable requesting components (e.g., Fig. 1 – The Hercules Framework – Calling Component) configured to execute one or more computer-executable target components (e.g., Fig. 1 – The Hercules Framework – Version1 … Version N) in the computerized system, a method of automatically providing a

computer-executable requesting component with access to an automatically determined version of a computer-executable target component upon request (e.g., Abstract, 2nd Par. - … to safely and reliably deploy and evolve component-based systems by executing and controlling multiple versions of software components at run-time …; Fig. 1 – The Hercules Framework – Arbiter and Constraint Evaluator; Sec. 3.1 – The component development process, Last Par. - … automated decisions can be made … that do not require human intervention), comprising the acts of:

- receiving one or more requests from one or more requesting components for access by the one or more requesting components of one or more target components, wherein each request includes an indication of the lowest possible version of the target component that the requesting component can accept (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration);
- upon receiving the one or more requests, identifying a versioning policy for each of the requested target components (e.g., P. 196, Left-Col., 1st Par. - … The Arbiter invokes each of the component versions when a request arrives, and sends the selected result back to the system …);
- automatically determining from the identified versioning policy that the requested target component is a platform component or a library component; (e.g., P. 196, Left-Col., 1st Par. - … To select a result, the Arbiter uses a Constraint Evaluator

(CE), providing the invocation ... to the CE. The CE evaluates the formal specifications of each version's addressed sub-domain, and decides which version of the component … The Arbiter then selects this result to send back to the external system …)

- automatically providing the one or more requesting component with access to an appropriate version of the one or more target components on a differential basis from one target component to the next, wherein:
    - if the requested target component is a platform component, the requesting component is automatically provided only the most recent servicing of the target component that is at least as recent as the lowest possible version of the target component specified by the requesting component (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration); and
    - if the requested target component is a library component, the requesting component is provided only a version of the target component that is specified by the requesting component (e.g., P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...).

5. **As to claim 4** (Currently Amended) (incorporating the rejection in claim 1), Cook discloses the method, wherein the target component is a platform component, the method further comprising an act of identifying a more recent version of the target component in response to a request for an earlier version of the target even though the more recent version and the earlier version are both accessible to the computerized system comprises identifying a more recent version of a platform component even though an earlier version of the platform component remained on the system when the more recent version was received at the computerized system (e.g., Sec. P. 195, Left-Col., 2nd Par., 2nd Bullet – Install the new version, but keep the old version(s) on-line as well)

6. **As to claim 5** (Previously Presented) (incorporating the rejection in claim 1), Cook discloses the method wherein further comprising an act of identifying the versioning policy of the specified lowest possible version of the target component when the specified lowest possible version of the target component is added to the computerized system (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration)

7. **As to claim 6** (Currently Amended) (incorporating the rejection in claim 1), Cook discloses the method wherein further comprising an act of storing, in the requesting

component, version information that identifies the specified lowest possible version of the target component in the requesting component when the requesting component is one or more of compiled, configured, installed, and run on the computerized system (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration)

8. **As to claim 7** (Currently Amended) (incorporating the rejection in claim 1), Cook discloses the method further comprising:

- identifying one or more requesting components that are able to access a prior version of the target component (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration);
- identifying that none of the one or more requesting components are configured to request the prior version of the target component (e.g., P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...); and
- automatically deleting the prior version of the target component (e.g., P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter

and only the newest version will be invoked ... the older component versions can be removed from the system ...)

9. **As to claim 8** (Previously Presented) (incorporating the rejection in claim 1), Cook discloses the method wherein the request further includes a request for a specific version of the target component, wherein the requested specific version is different from the lowest possible version of the target component (e.g., Sec. P. 195, Left-Col., 2nd Par., 2nd Bullet – Install the new version, but keep the old version(s) on-line as well)

10. **As to claim 9** (Previously Presented) (incorporating the rejection in claim 8), Cook discloses the method wherein the automatically determined appropriate version of the target component is different from the requested specific version of the target component that was requested (e.g., Sec. P. 195, Left-Col., 2nd Par., 2nd Bullet – Install the new version, but keep the old version(s) on-line as well)

11. **As to claim 10** (Previously Presented) (incorporating the rejection in claim 1), Cook discloses further comprising receiving a plurality of new versions of the target component, wherein each of the new versions of the target component is associated with a different versioning policy (e.g., P. 195, Left-Col., 1st Par. - … treat each component versions as a special multi-versioned system)

12. **As to claim 11** (Previously Presented) (incorporating the rejection in claim 9), Cook discloses the method further comprising determining the appropriate version of

the target component from among the specified lowest possible version of the target component and each of the plurality of new versions of the target component when the plurality of new versions of the target component (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration)

13. **As to claim 12** (Original) (incorporating the rejection in claim 1), Cook discloses the method wherein the versioning policy (e.g., Sec. 3.2 – Formal specification of the sub-domain - ... the sub-domain constraint specifications that can be used to determine which version is authoritative ...) is inserted into computer-executable instructions in the target component prior to one of installing, configuring, and executing the target component on the computerized system (e.g., P. 195, Left-Col., 1st Par. - … treat each component versions as a special multi-versioned system)

14. **As to claim 13** (Previously Presented) (incorporating the rejection in claim 1), Cook discloses the method wherein the versioning policy is further identified in a plurality of version of the target component on the computerized system (e.g., P. 195, Left-Col., 1st Par. - … treat each component versions as a special multi-versioned system)

15. **As to claim 14** (Previously Presented) (incorporating the rejection in claim 12), Cook discloses the method wherein each versioning policy in each version of the target component identifies a specific version of the requesting component configured to access that target component (e.g., P. 195, Left-Col., 1st Par. - … treat each component versions as a special multi-versioned system)

16. **As to claim 15** (Original) (incorporating the rejection in claim 1), Cook discloses the method further comprising identifying a component scope that is associated with the target component (e.g., P. 197, Right-Col., 1st Par. - … only domains over which value range …)

17. **As to claim 16** (Currently Amended) (incorporating the rejection in claim 1), Cook discloses the method wherein appropriate version of the target component is further automatically determined based on the identified component scope associated with the target component in addition to a determination of the lowest possible version that can be accepted (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration)

18. **As to claim 17** (Previously Presented) (incorporating the rejection in claim 15), Cook discloses the method wherein the identified component scope specifies that

access to the specified version of the target component is provided differently from the lowest possible version of the target component in one of a machine level, a process level, or a sub-process level (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration)

19. **As to claim 20** (Currently Amended), Cook discloses in a computerized system that includes one or more computer-executable program components including one or more computer-executable requesting components (e.g., Fig. 1 – The Hercules Framework – Calling Component) that can request to access one or more computer-executable target components (e.g., Fig. 1 – The Hercules Framework – Version1 … Version N) in the computerized system, a method of automatically providing a computer-executable requesting component with access to an automatically determined version of a target component (e.g., Abstract, 2nd Par. - … to safely and reliably deploy and evolve component-based systems by executing and controlling multiple versions of software components at run-time …; Fig. 1 – The Hercules Framework – Arbiter and Constraint Evaluator; Sec. 3.1 – The component development process, Last Par. - … automated decisions can be made … that do not require human intervention), comprising:

- receiving one or more requests from one or more requesting components for access by the one or more requesting components of one or more target

components, wherein each request includes an indication of the lowest possible version of the target component that the requesting component can accept (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration)

- a step for, upon receiving the request form the requesting component, automatically determining an appropriate version of the requested target component based on a versioning policy corresponding to the requested target component; and automatically allowing access to an appropriate version of the requested target component on a differential basis based on whether the requested target component is a platform component or a library component, such that the requesting component accesses the appropriate target component as it has been configured to do so, and such that the requesting component does not fail when requesting access to a component that has been upgraded (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration; P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...)

20. **As to claim 21** (Currently Amended) (incorporating the rejection in claim 20), Cook discloses the method wherein the step for allowing access to an appropriate version of the requested target component comprises the corresponding acts of:

- upon receiving the one or more requests, identifying a versioning policy for each of the requested target components (e.g., P. 196, Left-Col., 1st Par. - … The Arbiter invokes each of the component versions when a request arrives, and sends the selected result back to the system …);
- automatically determining from the identified versioning policy that the requested target component is a platform component or a library component (e.g., P. 196, Left-Col., 2nd Par. – The Arbiter also contains component management facilities, for dynamically adding and removing versions …);
- automatically providing the one or more requesting components with access to an appropriate version of the one or more target components on a differential basis from one target component to the next, wherein:
    - if the requested target component is a platform component, the requesting component is automatically provided only the most recent servicing of the target component that is at least as recent as the lowest possible version of the target component specified by the requesting component (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration); and

- if the requested target component is a library component, the requesting component is provided only a version of the target component that is specified by the requesting component (e.g., P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...)

21. **As to claim 22** (Currently Amended), Cook discloses in a computerized system that includes one or more program components including one or more requesting components (e.g., Fig. 1 – The Hercules Framework – Calling Component) that can request to access one or more target components (e.g., Fig. 1 – The Hercules Framework – Version1 … Version N) in the computerized system, a method of automatically managing access of one or more versions of computer executable target component (e.g., Abstract, 2nd Par. - … to safely and reliably deploy and evolve component-based systems by executing and controlling multiple versions of software components at run-time …; Fig. 1 – The Hercules Framework – Arbiter and Constraint Evaluator; Sec. 3.1 – The component development process, Last Par. - … automated decisions can be made … that do not require human intervention) such that a computer-executable requesting component that accesses the target component continues to operate effectively after the target component has been upgraded with newer versions (e.g., Fig. 2 – The component upgrading process) thereof, comprising the acts of:

- identifying that one ore more requesting components are configured to execute a version of one ore more computer-executable target components (e.g., P. 196, Left-Col., 1st Par. - … The Arbiter invokes each of the component versions when a request arrives, and sends the selected result back to the system …);
- automatically identifying a versioning policy for each of the one or more target components (e.g., P. 196, Left-Col., 1st Par. - … To select a result, the Arbiter uses a Constraint Evaluator (CE), providing the invocation ... to the CE. The CE evaluates the formal specifications of each version's addressed sub-domain, and decides which version of the component … The Arbiter then selects this result to send back to the external system …);
- automatically determining for each of the one or more target components whether the target component is a platform component or a library component (e.g., P. 196, Left-Col., 2nd Par. – The Arbiter also contains component management facilities, for dynamically adding and removing versions …) ; and
- for each platform component automatically determining based on the corresponding versioning policy for each platform component to remove any of the available versions of the platform component that are earlier than the version for which any of the one or more requesting components are configured (e.g., P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...); and

- for each library component, determining based on the corresponding versioning policy to maintain all new versions of the target component, all the existing versions of the target component, and all of the previously installed version of the target component in the system at the same time (e.g., Sec. P. 195, Left-Col., 2nd Par., 2nd Bullet – Install the new version, but keep the old version(s) on-line as well)

22. **As to claim 26** (Currently Amended), Cook discloses in a computerized system including one or more requesting components (e.g., Fig. 1 – The Hercules Framework – Calling Component) that are configured to access one or more target components (e.g., Fig. 1 – The Hercules Framework – Version1 … Version N) in the computerized system, a computer storage product having computer-executable instructions stored thereon that, when executed, cause one or more processors in the computerized system to execute a method of automatically providing a computer-executable requesting component with access to an automatically determined version of a computer-executable target component upon request (e.g., Abstract, 2nd Par. - … to safely and reliably deploy and evolve component-based systems by executing and controlling multiple versions of software components at run-time …; Fig. 1 – The Hercules Framework – Arbiter and Constraint Evaluator; Sec. 3.1 – The component development process, Last Par. - … automated decisions can be made … that do not require human intervention), comprising the acts of:

- receiving one or more requests from one or more requesting components for access by the one or more requesting components of one or more target components, wherein each request includes an indication of the lowest possible version of the target component that the requesting component can accept (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration);
- upon receiving the one or more requests (e.g., P. 196, Left-Col., 1st Par. - … The Arbiter invokes each of the component versions when a request arrives, and sends the selected result back to the system …), identifying a versioning policy for each of the requested target components;
- automatically determining from the identified versioning policy that the requested target component is a platform component or a library component (e.g., P. 196, Left-Col., 1st Par. - … To select a result, the Arbiter uses a Constraint Evaluator (CE), providing the invocation ... to the CE. The CE evaluates the formal specifications of each version's addressed sub-domain, and decides which version of the component … The Arbiter then selects this result to send back to the external system …)
- automatically providing the one or more requesting components with access to an appropriate version of the one or more target components on a differential basis for each target component, wherein:

- if the requested target component is a platform component, the requesting component is automatically provided only the most recent servicing of the target component that is at least as recent as the lowest possible version of the target component specified by the requesting component (e.g., Sec. 1 – Introduction, 1st Par. - … knowing what changes were made between successive versions of components, and knowing which versions of which components are compatible and can be composed into a testable system configuration); and
- if the requested target component is a library component, the requesting component is provided only a version of the target component that is specified by the requesting component (e.g., P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...)

23. **As to claim 27** (Currently Amended), Cook discloses in a computerized system including one or more requesting components that are configured to access one or more target components in the computerized system, a computer program storage product having computer-executable instructions stored thereon that, when executed, cause one or more processors in the computerized system to execute a method of automatically managing access of one or more versions of computer-executable target

component (e.g.,) such that a computer-executable requesting component (e.g., Fig. 1 – The Hercules Framework – Calling Component) that accesses the computer-executable target component (e.g., Fig. 1 – The Hercules Framework – Version1 … Version N) continues to operate effectively after the target component has been upgraded with newer versions thereof (e.g., Abstract, 2nd Par. - … to safely and reliably deploy and evolve component-based systems by executing and controlling multiple versions of software components at run-time …; Fig. 1 – The Hercules Framework – Arbiter and Constraint Evaluator; Sec. 3.1 – The component development process, Last Par. - … automated decisions can be made … that do not require human intervention), comprising the acts of:

1. identifying that one or more requesting components are configured to execute a version of one ore more computer-executable target components (e.g.,);

2. automatically identifying a versioning policy for each of the one or more target components (e.g., P. 196, Left-Col., 1st Par. - … The Arbiter invokes each of the component versions when a request arrives, and sends the selected result back to the system …);

3. automatically determining for each of the one or more target components whether the target component is a platform component or a library component (e.g., P. 196, Left-Col., 2nd Par. – The Arbiter also contains component management facilities, for dynamically adding and removing versions …); and

4. for each platform component automatically determining based on the corresponding versioning policy for each platform component to remove any of the

available versions of the platform component that are earlier than the version for which any of the one or more requesting components are configured (e.g., P. 198, Left-Col., 2nd Par. – 3rd Par. - … older versions cannot be invoked by the Arbiter and only the newest version will be invoked ... the older component versions can be removed from the system ...); and

5. for each library component, determining based on the corresponding versioning policy to maintain all new versions of the target component, all the existing versions of the target component, and all of the previously installed version of the target components in the system at the same time (e.g., Sec. P. 195, Left-Col., 2nd Par., 2nd Bullet – Install the new version, but keep the old version(s) on-line as well)

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 18-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cook in view of Steven Pratschner (*Simplifying Deployment and Solving DLL Hell with the .NET Framework™, Nov. 2001, Microsoft Corporation™, pp. 1-12*) (hereinafter 'Pratschner')

24. **As to claim 18** (Previously Presented) (incorporating the rejection in claim 1), Cook does not explicitly disclose the method further comprising identifying a servicing value associated with the requested target component.

However, in an analogous art of *Simplifying Deployment and Solving DLL Hell with the .NET Framework*™, Pratschner discloses the method wherein the requested target component is a library component, the method further comprising identifying a servicing value associated with the requested target component (e.g., P. 8, Sec. of "Custom Version Policy", the vendor of a shared assembly may have shipped a service release to an existing assembly and would like all applications to being using the service release instead of the original version; these scenarios and others are supported in the .NET Framework™ through version policies; P. 11, Sec. of "Deployment" – the .NET Framework™ provides extensive code download services are integration with Windows Installer).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Pratschner into the Cook system to further provide the method wherein the requested target component is a library component, the method further comprising identifying a servicing value associated with the requested target component in Cook system.

The motivation is that it would further enhance the Cook's system by taking, advancing and/or incorporating Pratschner's system which offers significant advantages that the CLR records version information between pieces of an application and uses that information at run time to ensure that the proper version of a dependency is loaded;

version policies can be used by both application developers and administrators to provide some flexibility in choosing which version of a given assembly is loaded as once suggested by Pratschner (e.g., P. 12, Sec. of "Summary")

25. **As to claim 19** (Previously Presented) (incorporating the rejection in claim 18), Pratschner discloses the method wherein identifying an appropriate version of the target component comprises identifying an updated servicing of a target component (e.g., P. 7, Sec. of "Version Policy" – including "Default Version Policy", "Custom Version Policy", and Version Policy Levels"; P. 8, Sec. of "Custom Version Policy", the vendor of a shared assembly may have shipped a service release to an existing assembly and wou8ld like all applications to being using the service release instead of the original version; these scenarios and others are supported in the .NET Framework™ through version policies; P. 11, Sec. of "Deployment" – the .NET Framework™ provides extensive code download services are integration with Windows Installer)

26. Claim 23 is rejected under 35 U.S.C. 103(a) as being unpatentable over Cook in view of Eisenbach et al. (*Managing the Evolution of .NET™ Programs, 2003, IFIP International Federation for Information Processing 2003* (hereinafter 'Eisenbach')

As to claim 23 (Currently Amended) (incorporating the rejection in claim 22), Cook does not explicitly disclose the method wherein each target component includes a versioning value and a servicing value, the method further comprising:

- receiving an updated servicing of the existing version of one of the target component over a network from a network service provider;
- automatically overwriting the target component, wherein the existing version of the target component reflects the versioning value and a new servicing value;

However, in an analogous art of *Managing the Evolution of .NET™ Programs*, Eisenbach discloses the method wherein each target component includes a versioning value and a servicing value, the method further comprising:

- receiving an updated servicing of the existing version of one of the target component over a network from a network service provider (e.g., Fig. 4 – Distributed Dejavue.NET architecture, element of 'Distributed Server'); and
- automatically overwriting the target component, wherein the existing version of the target component reflects the versioning value and a new servicing value (e.g., P. 186, 4th Par. – The design of the .NET assembly reveals some details about how this is accomplished, with each assembly carrying versioning information structured into four fields – Major, Minor (e.g., servicing value), Revision and Build. This allows many versions of the same DLL to co-exist in the system)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Eisenbach into the Cook's system to further provide the method wherein each target component includes a versioning value and a servicing value, the method further comprising:

- receiving an updated servicing of the existing version of one of the target component over a network from a network service provider;
- automatically overwriting the target component, wherein the existing version of the target component reflects the versioning value and a new servicing value.

The motivation is that it would further enhance the Cook's system by taking, advancing and/or incorporating the Eisenbach's system which offers significant advantages that a formal model was developed to assist in understanding the .NET mechanism and in describing our way of dealing with multiple versions and a tool was constructed to do so as once suggested by Eisenbach (e.g., Abstract, 2nd Par.)

Conclusion

27. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192
June 20, 2008

/Tuan Q. Dam/

Supervisory Patent Examiner, Art Unit 2192